ME426: Applied Computational Methods in Mechanical Sciences

# Submitted By: Himanshu Kumar (16ME234) August 19, 2019

# Assignment 3

PROBLEM STATEMENT:

Solving the linear system AX=B by Jacobi, Gauss-Siedel and SOR methods, with an error limit of 0.0001.

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | -1 | 2 | 0 |
| -1 | 11 | -1 | 3 |
| 2 | -1 | 10 | -1 |
| 0 | 3 | -1 | 8 |

|  |
| --- |
| 6 |
| 25 |
| -11 |
| 15 |

A = B =

Python Code:

import time

a= [[10,-1,2,0],

[-1,11,-1,3],

[2,-1,10,-1],

[0,3,-1,8]]

b= [6,25,-11,15]

guess = [1,1,1,1]

lim\_err = 0.0001

def diagonal\_dominance(a):

n=len(a)

dom=0

for i in range(n):

dom=0

for j in range(n):

if(i!=j and abs(a[i][i])>abs(a[i][j])):

dom=dom+1;

if(dom == (n-1)):

print("\ndiagionally dominant")

return(1)

else:

print("checking for next row")

print("Not diagionally dominant, needs reordering")

return(0)

def jacobi(a,b,guess,lim\_err):

#check diagonal dominance:

if(not(diagonal\_dominance(a))):

print("NA\_returning 0 ")

return(0)

n=len(a)

#forming equation matrix

z=[[a[i][j] for i in range(n)] for j in range(n)]

for i in range(n):

z[i][i] = 0

x=guess[:]

x\_next=[0]\*n

err = [0]\*n

iteration = 0

while(1):

#calculating iteration with error

max\_err=0

for i in range(n):

s = sum(z[i][j]\*x[j] for j in range(n))

x\_next[i]=(b[i]-s)/a[i][i]

#error

err[i] = ((x[i]-x\_next[i])/(x\_next[i]))

if(err[i]<0):

err[i] = (-1)\*err[i]

if(max\_err<err[i]):

max\_err=err[i]

iteration = iteration+1

#print("\n\nError is :",err)

#print("x :",x)

#print("x\_next :",x\_next)

x=x\_next[:]

if(max\_err<lim\_err):

break

print("\n\nIterations:",iteration)

print("solution:",x)

print("initial guess was:",guess)

print ("\n CPU time: ", time.process\_time(),'s')

return(x)

def gauss\_siedel(a,b,guess,lim\_err):

#check diagonal dominance:

if(not(diagonal\_dominance(a))):

print("NA\_returning 0 ")

return(0)

n=len(a)

#forming equation matrix

z=[[a[i][j] for i in range(n)] for j in range(n)]

for i in range(n):

z[i][i] = 0

x=guess[:]

x\_prev = x[:]

err = [0]\*n

iteration = 0

while(1):

#calculating iteration with error

max\_err=0

x\_prev = x[:]

for i in range(n):

s = sum(z[i][j]\*x[j] for j in range(n))

x[i]=(b[i]-s)/a[i][i]

#error

err[i] = ((x[i]-x\_prev[i])/(x[i]))

if(err[i]<0):

err[i] = (-1)\*err[i]

if(max\_err<err[i]):

max\_err=err[i]

#print("\n\nError is :",err)

#print("x\_prev :",x\_prev)

#print("x :",x)

iteration = iteration+1

if(max\_err<lim\_err):

break

print("\n\nIterations:",iteration)

print("solution:",x)

print("initial guess was:",guess)

print ("\n CPU time: ", time.process\_time(),'s')

return(x)

def sor(a,b,guess,lim\_err):

#check diagonal dominance:

if(not(diagonal\_dominance(a))):

print("NA\_returning 0 ")

return(0)

n=len(a)

#forming equation matrix

z=[[a[i][j] for i in range(n)] for j in range(n)]

for i in range(n):

z[i][i] = 0

result = []

w\_min\_iterations =(1,0)

w=1.0

while(w<2):

result,iters = sor\_w(a,b,w,z,n,guess,lim\_err)

print()

if(w-1):

if(w\_min\_iterations[1]> iters):

w\_min\_iterations = (w,iters)

else:

break

else:

w\_min\_iterations = (w,iters)

w=w+0.05

print("initial guess was:",guess)

print("\nOptimum w is:",w\_min\_iterations[0])

return(result)

def sor\_w(a,b,w,z,n,guess,lim\_err):

x=guess[:]

x\_prev = x[:]

err = [0]\*n

iteration = 0

while(1):

#calculating iteration with error

max\_err=0

x\_prev = x[:]

for i in range(n):

s = sum(z[i][j]\*x[j] for j in range(n))

x[i] = x\_prev[i]\*(1-w) +w\*((b[i]-s)/a[i][i])

#error

err[i] = ((x[i]-x\_prev[i])/(x[i]))

if(err[i]<0):

err[i] = (-1)\*err[i]

if(max\_err<err[i]):

max\_err=err[i]

#print("\n\nError is :",err)

#print("x\_prev :",x\_prev)

#print("x :",x)

iteration = iteration+1

if(max\_err<lim\_err):

break

print("\n\nIterations for sor:",iteration)

print("solution:",x)

print ("\n CPU time: ", time.process\_time(),'s')

return(x,iteration)

#sol=gauss\_siedel(a,b,guess,lim\_err)

#sol=jacobi(a,b,guess,lim\_err)

sol=sor(a,b,guess,lim\_err)

RESULT:

solution: [0.9999740731063248, 1.9999983621762443, -0.9999934260075907, 0.9999991174981038]

Iterations for sor: 5

Optimum w is: (1.05)

CPU TIME:

SOR:

CPU time: 0.15625 s

JACOBI:

CPU time: 0.265625 s

GAUSS SIEDEL:

CPU time: 0.25 s